1-1-2006

Fitness evaluation for structural optimisation genetic algorithms using neural networks

Koren Ward
University of Wollongong, koren@uow.edu.au

Timothy J. McCarthy
University of Wollongong, timmc@uow.edu.au

Fitness evaluation for structural optimisation genetic algorithms using neural networks

Abstract

This paper relates to the optimisation of structural design using Genetic Algorithms (GAs) and presents an improved method for determining the fitness of genetic codes that represent possible design solutions by using a neural network to generalize fitness. Two problems that often impede design optimization using genetic algorithms are expensive fitness evaluation and high epistasis. In this paper we show that by using a neural network as a fitness approximator, optimal solutions to certain design problems can be achieved in significantly less generations and with considerably less fitness evaluations.

Keywords

Fitness, evaluation, for, structural, optimisation, genetic, algorithms, using, neural, networks

Disciplines

Physical Sciences and Mathematics

Publication Details

Ward, K. & McCarthy, T. J. (2006). Fitness evaluation for structural optimisation genetic algorithms using neural networks. International Conference on Engineering Computational Technology (pp. 1-11). Stirling, UK: Civil Comp Press Ltd.

Proceedings of the Fifth International Conference on Engineering Computational Technology, Las Palmas de Granaria - Spain, 12-15 Sept. 2006

Fitness Evaluation for Structural Optimisation Genetic Algorithms Using Neural Networks

K.Ward† and T.J. McCarthy‡

†*School of Information Technology and Computer Science,*
‡*School of Civil Mining and Environmental Engineering,*
University of Wollongong, Australia

Abstract

This paper relates to the optimisation of structural design using Genetic Algorithms (GAs) and presents an improved method for determining the fitness of genetic codes that represent possible design solutions by using a neural network to generalize fitness. Two problems that often impede design optimization using genetic algorithms are expensive fitness evaluation and high epistasis. In this paper we show that by using a neural network as a fitness approximator, optimal solutions to certain design problems can be achieved in significantly less generations and with considerably less fitness evaluations.

Keywords: genetic algorithm, neural network, epistasis, fitness classifier, structural optimisation, truss.

1 Introduction

Two significant problems that often hinder design optimisation using genetic algorithms are expensive fitness evaluation and high epistasis. Expensive fitness evaluation results in slow evolution and occurs when it is computationally expensive to test the effectiveness of possible design solutions using an objective function. High epistasis occurs when certain genes lose their significance or value when other genes change. Consequently, when a fit genetic code has an important gene changed this can have a dramatic effect on the fitness of that genetic code. Often the reduction in fitness results in failure of the genetic code being selected for reproduction and inclusion in the next generation. This loss of evolved genetic information can result in the solution taking considerable time to be discovered.

Most attempts at overcoming expensive fitness evaluations involve saving the fitness evaluations in a file or memory so they can subsequently be looked up, instead of being evaluated again, if the same genetic code occurs more than once in

the same population, or again in a later generation [1]. Although saving fitness evaluations for later reference can provide a cost saving, many fitness evaluations still have to be done, making some problems unviable for GAs, particularly if the genetic code is large and the objective function expensive.

Reducing high epistasis is usually done by representing the problem in a different manner or with different parameters. Sometimes, placing dependant genes next to each other in the genetic code can assist in preventing these genes becoming separated by the crossover operation. However, design problems with high epistasis generally remain difficult to solve with GAs or by other means.

To overcome these two fundamental problems with GAs we have been experimenting with back-propagation neural that are trained to recognise the fitness of genetic codes. Training the neural network is achieved by using training patterns comprised of genetic codes and their fitness which is obtained from the fitness function or memory. Although, this still requires fitness evaluations to be done, our experiments have shown that only a subset of the population is needed to train the neural network to classify fitness sufficiently for evolution to progress. This can result in a considerable cost saving when it is expensive and time consuming to perform fitness evaluations.

The neural network produces an estimate of the fitness of genetic codes, based on its architecture and training. Our experiments have shown that the neural network's ability to generalise enables substantial portions of fit gene strings to be identified and appropriately awarded fitness even if the whole genetic code has not occurred before. Furthermore, when important genes change, this may influence the significance of other genes in the genetic code. With the neural network, this does not have such a disruptive effect on the genetic code's classified fitness and can allow significant fit gene strings to remain represented in the population in certain problems with high epistasis.

To demonstrate this effect, we provide experimental results involving a classical design optimisation of a 10-bar indeterminate steel truss with genetic algorithms [2, 3, 4]. We compare a traditional GA with the same GA equipped with a neural network for generalizing and classifying the fitness of genetic codes. Our results show that the GA equipped with the neural network is not only able to find optimal solutions with considerably less fitness evaluations, it is also able to discover optimal solutions in significantly fewer generations than the traditional GA.

2 Design Optimization using Genetic Algorithms

When it comes to finding optimal solutions to difficult problems in structural design, analytical methods are often limited to near approximations due to the complexity of the real-world problem. Often the objective function and constrains are nonlinear and difficult to solve mathematically. Alternatively, various iterative optimization methods have been applied to different fields in science and engineering. These

methods include simulated annealing like Moh and Chiang [5], gradient-based methods such as the work by Taylor and Rossow [6], Kirsch [7], and genetic algorithms, e.g. (Atrek [2], Goldberg [8] and Turkkan [4]).

Genetic algorithms simulate the principles of natural selection and survival of the fittest. Namely, the "fitter" members of a population of possible solutions are more likely to survive and in turn produce fit offspring comprised of their own genes. To implement this on a computer for solving hard problems, the design is firstly parameterized into a gene string and a population of random solutions is generated and evaluated using the objective function. Fitter members of the population are then chosen in pairs and their genes are broken apart and recombined (crossed over) to form offspring comprising a new population as shown in Figure 1. By combining different strong characteristics in this way, fitter offspring are likely to be produced in the new population. Also, occasionally genes are mutated after crossover is performed to assist in the discovery of new stronger genes.

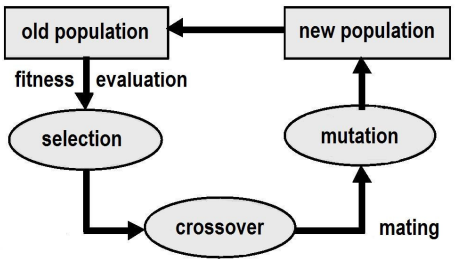

Figure 1: Basic genetic algorithm cycle.

Genetic algorithms have been demonstrated to be effective global optimizers which can often perform better than conventional optimization algorithms, particularly on problems which are discontinuous, non-differential, multi-modal, noisy or not well-defined. Such problems are often encountered in engineering and experimental designs [9]. Genetic algorithms are also well suited for multi-criteria optimization problems where the solution may be a compromise between multiple objectives, for example maximum strength versus minimum cost. The process of optimizing a collection of objective functions is often referred to as multi-objective optimization [10].

Despite considerable success being achieved in applying genetic algorithms to many real-world problems, genetic algorithms require a large number of fitness evaluations in order to discover optimal or near optimal solutions. This expense can make genetic algorithms infeasible on problems where the objective function is computationally expensive or difficult to simulate in a computer. Furthermore, the process of crossing over genes of population members to form offspring for the next generation can have a disruptive effect when genes which depend on each other for fitness become separated by this process.

To overcome the difficulty posed by expensive objective functions in design optimization problems we began experimenting with neural networks for approximating the fitness of population members. We found that not only did the addition of the neural network fitness approximator reduce the number of fitness evaluations needed to find near optimal solutions, but on some problems, the genetic algorithm equipped with the fitness approximator would discover near optimal solutions faster than the conventional genetic algorithm using fitness measures derived directly from the objective function. In the following sections we demonstrate this effect through the design optimisation of a 10-bar indeterminate steel truss with a genetic algorithms using a neural network fitness approximator.

3 Truss Optimization with Genetic Algorithms

Truss optimization can be divided into three main categories: 1. sizing, 2. configuration and 3. topology. Sizing optimization is where the cross-sectional areas of the truss members are design variables and the coordinates of the nodes are held constant [3, 4]. The aim is usually to minimize the weight of the structure while complying with certain constraints placed on stress and displacement. The sizing problem is made more difficult by restricting the choice of truss members to a set of available standard cross-sections, see [11].

Optimization of the configuration or topology of trusses aims to improve a given topology or configuration by minimizing an objective function subject to a number of constraints. The design variables are often the coordinates of key points in the boundary of the structure, see [12, 13, 14].

One classical truss sizing optimization problem often applied to genetic algorithms involves discovering the cross-section member areas of the 10-bar indeterminate steel truss shown in Figure 2. The objective here is to minimize the overall weight of the truss without over-stressing the structure or causing significant deflection, see [2, 8, 4]. Appendix A gives the material properties and constraints. Table 1 shows the cross-sectional areas of the 42 steel sections from which truss members can be fabricated. Since there are 10 design variables and 42 available shapes, the size of the search space is 42^{10} or approximately 10^{16}.

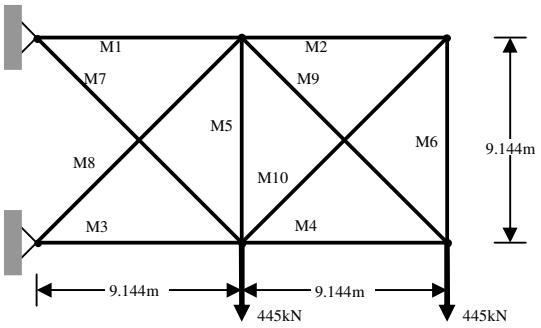

Figure 2: 10-bar indeterminate steel truss problem.

ID	Area	ID	Area	ID	Area
1	1045	15	2342	29	7419
2	1161	16	2477	30	8710
3	1284	17	2497	31	8968
4	1374	18	2503	32	9161
5	1535	19	2697	33	10000
6	1690	20	2723	34	10323
7	1697	21	2897	35	10903
8	1858	22	2961	36	12129
9	1890	23	3097	37	12839
10	1993	24	3206	38	14193
11	2019	25	3303	39	14774
12	2181	26	3703	40	17097
13	2239	27	4658	41	19355
14	2290	28	5142	42	21613

Table 1: Cross-sectional areas (mm^2) of steel bar.

To optimize a 10-bar truss, 3 measures of fitness need to be evaluated and considered to determine the overall fitness of the structure. These measures are mass, overstress and deflection which are expressed in Equations 1a, 1b and 1c respectively. Equation 1d describes a typical objective function used to evaluate the fitness of population members within the genetic algorithm [4].

$$M = \sum_{i=1}^{10} A_{j(i)} L \tag{1a}$$

$$S = \sum_{i=1}^{10} \delta_i (e^{g_{ij}} - 1) \tag{1b}$$

$$D = \sum_{i=1}^{12} \delta_k (e^{g_{2k}} - 1) \tag{1c}$$

where

$j(i) = 1, 2, \ldots 42$

$\delta = 1 \quad for\ g > 0$

or $\delta = 0 \quad otherwise$

and

$$\min W = M + P(S + D) \tag{1d}$$

where

P is a penalty coefficient.

The objective here is to minimize the fitness measure W. Normally, the penalty coefficient P is large so that population members that are overstressed or have excessive deflection are appropriated considerably worse fitness than population members that are structurally stable.

4 Fitness Estimation using a Neural Network

Recently, the increased use of evolutionary strategies for solving problems with expensive objective functions has lead to the development of a variety of fitness approximation techniques, see [15] for a survey. Generally, fitness approximators or surrogate objective functions are used in genetic algorithms in order to reduce the number of expensive fitness evaluations needed to find optimal solutions to various problems. One form of fitness approximator involves using a neural network to learn associations between genetic codes and their fitness, see [16] for examples. However, little has been mentioned on deploying neural network fitness approximators for improving the performance of genetic algorithms which can be achieved in certain circumstances.

An example of a GA where a neural network fitness approximator can perform better than the actual objective function (in terms of finding optimal solutions in less generations) is the 10-bar truss problem where the objective function has been modified so that structurally stable population members are assigned fitness proportional to their mass and structurally unstable population members (i.e. members that are over stressed or have excessive deflection) are assigned the maximum (worst possible) fitness. Namely:

$$\begin{array}{lll} & W = M & \text{for } S < S_t \text{ and } D < D_t \\ \text{or} & W = \max W & \text{otherwise} \end{array} \qquad (2)$$

where

S_t is the overstress threshold and
D_t is the deflection limit

In the following section we provide experimental results demonstrating some beneficial effects of using a neural network fitness approximator in a GA for fitness estimation and classification. The first experiment shows how near optimal solutions of the 10-bar truss problem can be found with considerably less fitness evaluations by using a neural network to estimate the fitness of population members. The second experiment shows how a neural network classifier can assist a GA to find optimal solutions in situations where the objective function makes it difficult to resolve part of the search space.

5 Experimental Results

Training a back-propagation neural network to learn the fitness of genetic codes can be done by using training data derived from the population where the inputs are comprised of the gene values of population members and the output is their associated fitness. However, before this can be done the input and output values must be normalized to comply with the requirements of the neural net's sigmoid activation functions.

To normalize the inputs we simply divided the each gene value by their maximum. However, we found using this approach to normalize the output (i.e. the fitness values) did not provide adequate resolution to differentiate the fitness of population members within specific generations. This problem was overcome by firstly squashing each component of the fitness (see Equation 1) with the sigmoid function and reducing the penalty coefficient. The sum of the fitness components was then applied to another sigmoid function to normalize the resulting fitness to between 0 and 1.

Training the neural network is done by taking a subset of the population, finding the fitness of each member by using the objective function and training the net with the resulting exemplars. To reduce the number of fitness evaluations done with the objective function the training data subset size is reduced for each generation. Namely:

$$n = p(1 - \frac{g}{\max g}) \tag{3}$$

where

p is the population size and
g is the generation number

Furthermore, a hash table is used to store and lookup evaluated fitness values to avoid having to evaluate the same gene vector multiple times with the objective function if it occurs more than once. Care needs to be taken to avoid over training the net. For our experiments we found 1500 training iterations per generation to be effective for a neural network with one hidden layer comprised of 8 nodes. For all trials the learning rate was set to 0.3 and the momentum was set to 0.2. For the GA settings, elite roulette wheel selection was used with replacement together with the linear crossover techniques. The population size was set to 400. The probability of crossover and mutation was set to 0.85 and 0.2 respectively.

To evaluate each population, members chosen to be in the training data subset were associated with their fitness from the objective function. All other members were either assigned their fitness from the hash table or from the neural net by classifying each gene vector. Figure 3 shows the best member fitness from each generation for both the GA equipped with the NN fitness approximator and a conventional GA with no fitness approximator. The values given in Figure 3 are taken from the average of 5 trial runs. The number of fitness evaluations performed by the objective function over the course of evolution can be seen in Figure 4. This represents an overall cost saving of approximately 70% in terms of the number of population members evaluated with the objective function, despite the result being produced in more generations. Figure 5 shows the solution produced by the GA equipped with the fitness approximator which compares well with the result produced by the conventional GA. The total mass of this truss is 2702kg. This compares with a best value in [4] of 2491kg after 2000 generations.

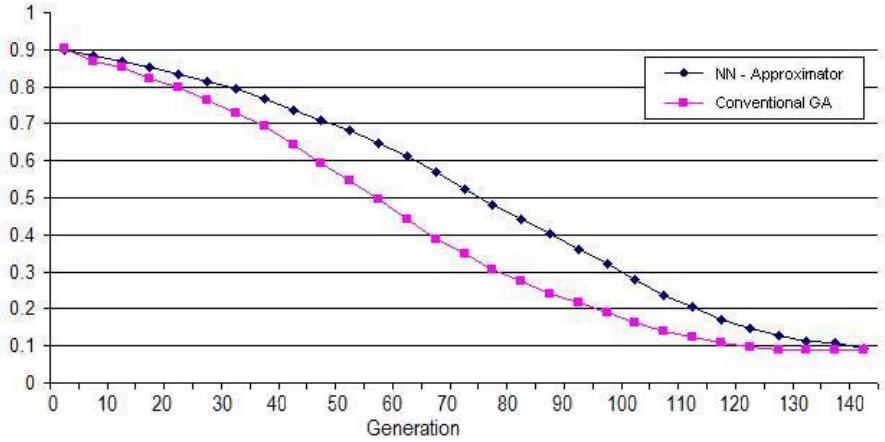

Figure 3: Best member fitness of over 140 generations.

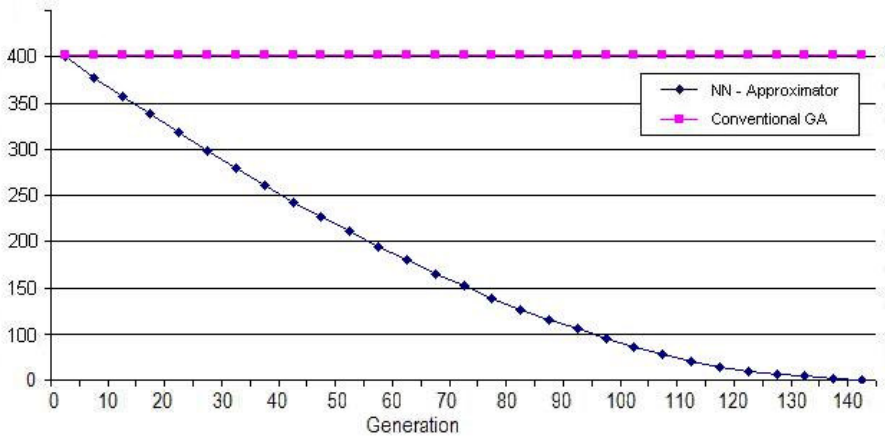

Figure 4: Number of fitness evaluations performed in each generation.

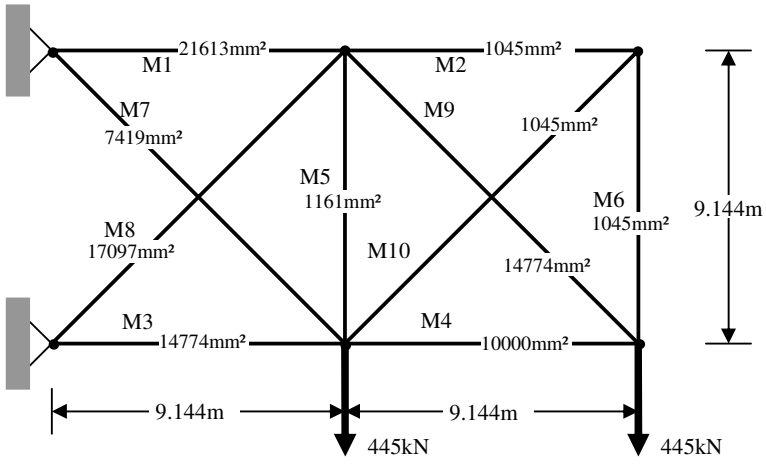

Figure 5: Final solution after 140 generations.

It is clear in Figure 3 that the number of generations required by the NN approximator is higher than the conventional GA. We made various adjustments to the neural network and GA parameters but were unable to get the GA equipped with the fitness approximator to find the solution in the same number of generations as the conventional GA. The reason for this was due to the neural network being unable to resolve slight variations in fitness that often occurred within some generations. This was mainly attributed to the amount of squashing applied to the deflection and overstress components of the fitness function, as explained in Section 3. To address this issue, we replaced the deflection and overstress squashing functions with binary threshold functions, (see Section 3, Equation 2). Furthermore, to prevent the initial population from becoming comprised entirely of members that happened to be overstressed and/or with excessive deflection, we set a maximum fitness constraint on the initial population so that only randomly generated members with fitness below 80% of the maximum were accepted. This measure did not significantly increase the time needed to generate the initial population and succeeded in producing an initial population comprised of members with considerable variation in mass, overstress and deflection characteristics.

Figure 6 shows the best member fitness from each generation for both the GA equipped with the fitness approximator and a conventional GA (again taken from the average of 5 trial runs). It is significant that the GA equipped with the fitness approximator is able to find near optimal solutions within less generations than the conventional GA. We believe this is due to the neural network's ability to generalize unknown or unstable gene vectors and return a measure similarity rather than the raw fitness value available from the fitness function.

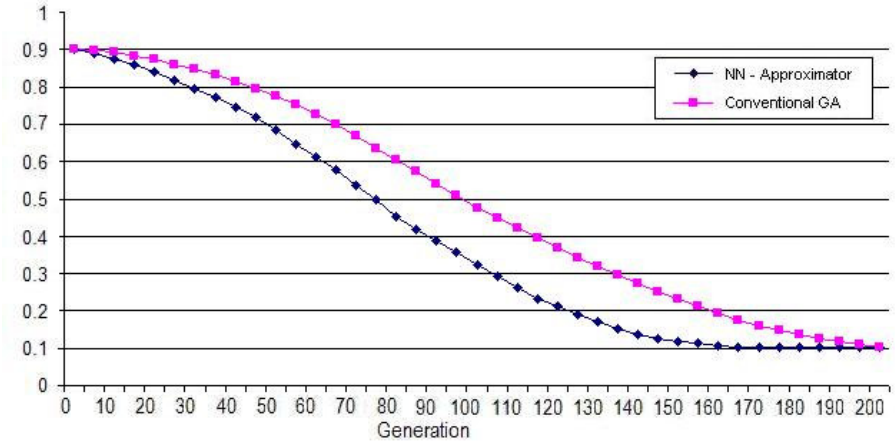

Figure 6: Fitness with binary penalty fitness function.

5 Conclusion

Although neural networks have previously been used for approximating the fitness of population members within genetic algorithms, this work has shown that neural network fitness approximators are not only able to be used to reduce the number of

fitness evaluation needed to find near optimal solution on GAs, but also can find the near optimal solutions in less generations under certain circumstances.

We demonstrated this by firstly using a GA equipped with a neural network fitness approximator to reduce the total number of fitness evaluations performed by the objective function for the classical 10-bar truss problem. We then demonstrated how the same GA equipped with the fitness approximator could find near optimal solutions in less generations than a traditional GA. This was achieved by altering the fitness function so that the GA was able to benefit from the neural network's ability to generalize the fitness of unstable designs.

We believe this result may have application on design problems where the only practical mean of evaluating the fitness of different designs is to indicate if changes made to a design either stand up to test conditions or fail.

References

[1] V.B. Gantovnik, C.M. Anderson-Cook, Z. Gurdal, and L.T. Watson. "A genetic algorithm with memory for mixed discrete-continuous design optimization". In "Computers & Structures", 81(20), 2003-2009, 2003.

[2] E. Atrek, "A theory for optimum design of skeletal structures." In "New Directions in Optimum Structural Design", E. Atrek, R. H. Gallagher, K. M. Ragsdell, and O. C. Zienkiewicz, eds., John Wiley and Sons, Chichester, 343-363. 1984.

[3] D. Goldberg, and M. Samtan, “Engineering optimization via genetic algorithms,” In "Proceeding of the 9th Conf. on Electronic Computations", ASCE, Birmingham, pp. 471-482, 1986.

[4] N. Turkkan, "Discrete optimization of structures using a floating-point genetic algorithm", In "Proceedings of the Annual Conference of the Canadian Society for Civil Engineering", Moncton, N.B., Canada, 2003.

[5] J. Moh and D. Chiang, “Improved simulated annealing search for structural optimization,” In "AIAA Journal", Vol. 38, pp. 1965-1973, 2000.

[6] J.E. Taylor and M.P. Rossow, “An optimal structural design using optimality criteria,” In "Advances in Engineering Science", 13th Annual Meeting, Hampton, VA, pp. 521-530, 1976.

[7] U. Kirsch, “Optimal design of trusses by approximate compatibility”, In "Computers and Structures", V 12, pp. 93-98, 1979.

[8] D. Goldberg, “Genetic algorithms in search, optimization and machine learning”, Addison-Wesley, 1989.

[9] H.P. Schwefel, "Evolution and optimum seeking", In "Sixth-Generation Computer Technology Series", Wiley, New York, 2005.

[10] R.T. Marler and J.S. Arora, “Review of multi-objective optimization: concepts and methods for engineering”, Technical Report Number ODL-01.01, University of Iowa, Optimial Design Laboratory, Iowa City, IA. 2003.

[11] S. Rajeev, C.S. Krishnamoorty, "Discrete optimization of structures using genetic algorithms". In "Journal of Structural Engineering ASCE", 118(5):123350, 1992.

[12] M. Papadrakakis, Y. Tsompanakis, E. Hinton and J. Sienz, “Advanced solution methods in topology optimization and shape sensitivity analysis”, In "I. Engrg. Comput." 13(5) 57-90. 1996.

[13] E. Hinton and J. Sienz, "Aspects of adaptive finite element analysis and structural optimization", In "B.H.V Topping and M. Papadrakakis, eds., "Advances in Structural Optimization" (CIVIL-COMP Press, Edinburgh) l-26, 1994.

[14] E. Ramm, K.U. Bletzinger, R. Reitinger and K. Maute, "The challenge of structural optimization", In B.H.V Topping and M. Papadrakakis, eds., "Advances in Structural Optimization" (CIVIL-COMP Press, Edinburgh) 27-52, 1994.

[15] J. Jin, "A comprehensive survey of fitness approximation in evolutionary computation", In "Soft Computing", 9:3–12 Springer-Verlag, 2005.

[16] M. Papadrakakis, N.D. Lagaos, and Y. Tsompanakis, “Structural optimisation using evolutionary strategies and neural networks”, In "Journal of Computer Methods in Applied Mechanics and Engineering", 156: 309-333, 1998.

Appendix A

The indeterminate 10 bar truss problem has been used as an optimization test using conventional techniques as well as GA solutions. It is an artificial problem constructed to have its optimum simultaneously governed by the deflection criterion and the stress criterion. The member sizes presented in Table 1 are metric equivalents of the AISC values. The modulus of elasticity used in the analysis is 68.947GPa (10000ksi), the deflection limit is 50.8mm (2 in) and the maximum allowable stress is 172.37 MPa (25ksi). The density of the material is 2770 kg/m^3.